DMB용 움직임 벡터 추출 프로그램 성능평가

김경현, 최덕영, 손승일

한신대학교 정보통신학과

Performance Evaluation of Motion Vector Prediction Program for DMB

Kyung Hyun Kim, Dug Young Choi, Seung Il Sonh

Dept. of Information and Communication HanShin University

E-mail : 52103kkh@hanmail.net

요 약

최근 들어 영상을 중심으로 여러 형태의 정보를 결합하여 저장하거나 전송하는 소위 '멀티미디어'가 전 세계적 열풍을 불러일으키고 있다. 디지털 기술에 바탕을 둔 멀티미디어 혁명은 멀티미디어 PC, 주문형 비디오(VOD), HDTV, 디지털 방송, DVD등 여러 가지 상품의 형태로 소비자들에게 선보이고 있다. 디지털 비디오 압축 기술은 이러한 멀티미디어 응용분야의 핵심이며, 압축 관련 국제 표준안 중 가장 최근에 발표된 것은 MPEG-4와 H.264가 있다.

이에 본 논문은 동영상 압축 기술의 핵심인 움직임 예측에 대하여 연구하고 최근 가장 활발히 연구되고 있는 DMB용 움직임 벡터 추출 프로그램을 구현하며 성능평가 할 것이다. 뿐만 아니라 이후 이를 바탕으로 움직임 벡터 추출 모듈을 VHDL로 구현한 후 성능평가의 기준으로 이용할 것이다.

1. 서론

최근 들어 영상을 중심으로 여러 형태의 정보를 결합하여 저장하거나 전송하는 소위 '멀티미디어'가 전 세계적 열풍을 불러일으키고 있다. 디지털 기술에 바탕을 둔 멀티미디어 혁명은 멀티미디어 PC, 주문형 비디오(VOD), HDTV, 디지털 방송, DVD등 여러 가지 상품의 형태로 소비자들에게 선보이고 있다. 디지털 비디오 압축 기술은 이러한 멀티미디어 응용분야의 핵심이며, 압축 관련 국제 표준안 중 가장 최근에 발표된 것은 MPEG-4와 H.264가 있다.

영상을 압축하는 방법에는 공간적 압축방법과, 시간적 압축방법 그리고 가변 길이로 부호화 할당하는 VLC(Variable Length Coding)방법이 있다. 공간적 압축방법은 화면 한 장 내에서의 상관관계를 이용하여 사람 눈에 민감하지 않은 고주파 성분을 제거함으로써 압축하는 방식이며 대표적인 것으로는 DCT 변환 코딩이 주로 사용되고 있다. 시간적인 압축 방법으로는 현재 화면과 다음 화면의 상관관계를 이용하여 움직임이 있는 부분만을 찾아 코팅하여 압축하는 방식이며 대표적으로 움직임 예측 및 보상이 쓰이고 있다. 끝으로 VLC방법으로는 공간적 압축방법과, 시간적 압축방법에 의해 변화된 정보를 정보의 확률적 통계에 의해 가변 길이로 부호화를 할당하는 방법으로 압축을 하며 허프만 코딩이나 산술 부호화 등이 사용되고 있다.

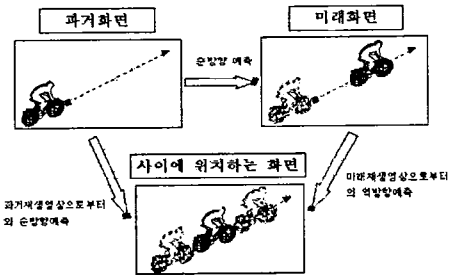
본 논문에서 연구하고자 하는 움직임 예측 및 보상 기능은 동영상 압축 표준안인 MPEG나 H.264에 가장 핵심적인 부분으로써 동영상 압축 부호화를 위한 전체 연산의 60%이상을 차지한다. 따라서 움직임 예측 및 보상 기능을 어떻게 구현하느냐에 따라 압축의 효율 및 그 성능이 좌우된다. 따라서 본 연구에서는 유연성을 강조하는 MPEG-4 보다는 효율성과 신뢰성을 강조하는 H.264를 표준으로 채택한 DMB용 움직임 예측 추출 프로그램을 효율적으로 구현하고 성능 평가 하였다.

2. 시간적 압축

영상 압축에서 말하는 시간적 압축이란 화면간의 상관관계를 이용한 압축을 말한다. 시간축으로 연속된 화면들은 주로 화면의 중앙부분에서 사람이나 물체의 움직임이 있기 때문에 움직임 보상 방법에서는 이러한 성질을 이용하여 시간축의 중복성을 제거한다. 즉 화면의 변하지 않은 부분이나 움직였다 하더라도 비슷한 부분을 바로 전 화면에서 가져와 채움으로써 전송해야 할 데이터량을 큰 폭으로 줄일 수 있다. 이렇게 화면사이에서 가장 비슷한 블록을 찾는 작업을 움직임 예측이라하며, 얼마만큼 움직였는가 하는 변위를 나타내는 것을 움직임 벡터라고 한다.

그림 1은 화면간의 상관관계를 나타내는 그림으로 MPEG이나 H.26x 에서는 화면을 3가지로 나

누는데 그림에서 과거화면 즉 첫 영상은 I-픽쳐라고 하며 미래화면은 P-픽쳐 라고 말한다. 그리고 사이에 위치하는 화면은 B-픽쳐라고 말한다. I-픽쳐는 시간적 압축을 사용하진 않고 P-픽쳐와 B-픽쳐만 시간적 압축을 사용하는데 P-픽쳐는 I-픽쳐를 기준으로 순방향 예측만 사용하고 B-픽쳐는 I-픽쳐와 B-픽쳐를 기준으로 순방향과 역방향 예측 모두를 사용한다[1][2].



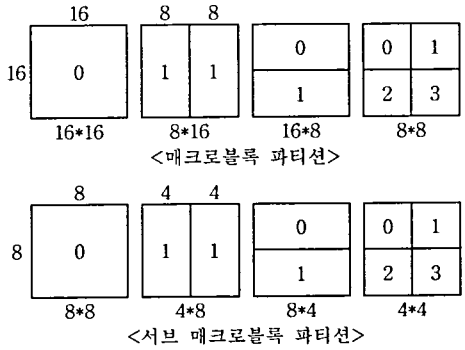
[그림 1] 화면간의 상관관계

3. H.264

H.264는 전기 전자 통신 분야 국제 표준화 기구 인 ITU-T산하 SG16 Q.6에서 레코멘데이션(Recommendation) H.263을 차세대 동영상 부호화 표준으로 제정하기 위해 1998년 1월 발족한 새로운 비디오 압축 기술 정의 프로젝트이다. ITU-T SG16 Q.6에서는 H.263 프로젝트와 병행해 H.26L 프로젝트를 진행하다. 점차적으로 H.26L 프로젝트에 비중을 두고 진행하였으며, 이러한 노력은 성공적으로 실행돼 MPEG과 공동 작업을 통하여 2003년도 6월에 FDIS(Final Draft for International Standard)를 거쳐서 표준화에 성공하였다. 2001년 12월 3일~7일에 걸쳐서 태국 파타야 에서 개최된 제 58회 MPEG 회의에서는 공식적으로 ISO/IEC JTC1 SC29 WG11과 JVT(Joint Video Team)의 발족을 선언하고, 서로의 협력 하에 H.26L 프로젝트를 진행시키기로 했다. 그 후 기존 VCEG과 MPEG 회원들의 협력으로 탄생된 JVT에서는 H.26L의 표준화를 계속하여 그 결과물로 탄생하게 될 국제 표준 동영상 부호기를 ISO와 ITU 각각 채택하기로 하였다. 이 결과로 인하여 ISO/IEC에서는 MPEG-4 Part10 AVC라는 이름으로 발표하였으며, ITU에서는 레코멘데이션 H.264라는 이름으로 공식 발표하였다.

H.264는 MPEG과 다르게 효율성과 신뢰성을 강조하였으며 기존의 MPEG보다 2배정도 좋은 효율을 보여주고 있다. H.264에는 특정한 기능을 지원하는 3가지의 프로파일 정의되어 있는데, 각각의 프로파일과 호환되기 위해서 인코더와 디코더에 요구되는 사항들이 정의되어 있다.

H.264의 움직임 보상을 보면 트리 구조 형식으로 되어 있는데 이는 하나의 16*16 매크로블록 파티션, 두 개의 16*8 파티션, 두 개의 8*16 파티션 또는 네 개의 8*8 파티션으로 움직임 보상될 수 있다. 8*8모드가 선택되면 매크로블록 내의 네 개의 8*8 서브 매크로블록은 각각 4가지 방법으로 다시 분할될 수 있다[3].



[그림 2] 매크로블록 파티션과 서브 매크로블록 파티션

H.264를 표준으로 채택한 DMB는 H.264의 3가지 프로파일 중 베이스 프로파일을 사용하며 베이스 프로파일은 I와 P 픽처만 지원해 준다.

4. 완전 탐색 블록 정합 방식

4.1 블록 정합

움직임 예측을 위해 가장 널리 사용되는 알고리즘은 블록 정합 알고리즘이다. 즉, 주어진 블록 내의 모든 픽셀에 대하여 변위를 구하고 그 중 가장 작은 변위를 나타내는 탐색 점의 값을 움직임 벡터로 추정하는 것이다. 현재 블록 정합 알고리즘에 대하여 활발히 연구되고 있으며 고속의 탐색을 위한 2차원 로그 탐색 알고리즘, 3단계 탐색 알고리즘, 교차 탐색 알고리즘 등 많은 고속 탐색 알고리즘이 연구되었다. 하지만 이와 같은 고속 탐색 알고리즘은 연산양은 적으나 의외의 경우가 발생하기 때문에 하드웨어로 구현하기에 용이하지 않다. 반면 완전 탐색 블록 정합 방식은 연산양은 많지만 규칙성을 가지고 있기 때문에 하드웨어 구현 시 용이하다.

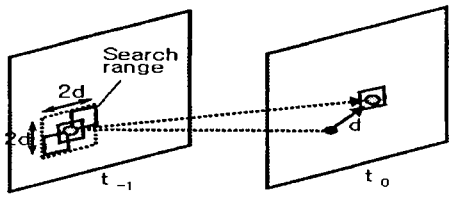
4.2 완전 탐색 블록 정합 방식

완전 탐색 블록정합 방식은 현재 화면의 기준 블록을 중심으로 이전 화면 안에 일정한 탐색범위를 만들어 이들 탐색 영역에 존재하는 모든 후보블록과 정합을 비교한다. 두 블록의 모든 화소에 대해 화소 간 차이 값의 절대치를 누적하여 그 값을 평균한 값으로부터 정합의 정도를 판단한다. 모든 후보블록에 대해 화소간 차이 절대치 누적의 평균값을 계산해야 하므로 엄청난 연산을 요구하는 반면에 다른 고속 탐색 블록정합에 비해 최소 예측 오차를 가지는 블록, 즉 최적의 예측

블록을 추출할 수 있다[2][4].

```
For (u=-p_h, u=p_h, u=u+1)
  For (v=-p_v, v=p_v, v=v+1)
   For (i=0, i=N-1, i=i+1)
    For (j=0, j=N-1, j=j+1)
      MAD(u, v) = MAD(u, v) + |R(i, j)-S(i+u, j+v)|
    End (j)
   End (i)
  if ( MAD(u, v) < MAD_champ )
  MAD_champ = MAD(u, v)
  MV_champ = (u, v)
 End (v)
End (u)
```

<식1> MAD를 이용한 완전탐색 블록정합 알고리즘



[그림 3] 완전 탐색 블록 정합 방식

그림 3는 완전탐색 블록 정합 방식의 그림으로 매크로 블록의 움직임 벡터 계산을 나타내 특정 위치의 이전영상 블록을 이동영상의 동일 위치에서 탐색한다. 이 과정을 통해 움직임 벡터 d를 추출 할 수 있다.

4.3 최소비용 함수

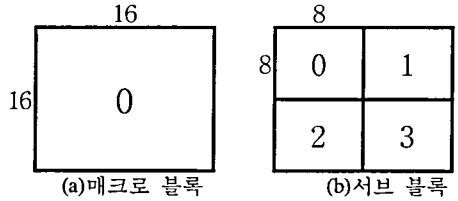
어떤 위치가 가장 비슷한 위치인지 결정하는 함수로는 MAD, MSD 함수를 많이 사용한다. 그 중에서 MAD는 제곱 연산이 없어 하드웨어로 설계하기에 용이하다. 또한, 이와 같은 최소비용 함수는 움직임 벡터의 존재 여부나 움직임 예측을 적용 할 것인지에 대한 기준 값이 되기도 한다. 식 2는 최소 비용함수를 보여주고 있다[5].

$$MSD(d) = \frac{1}{MN}\sum_{(x,y)\in R}(f(x+d_x,y+d_x,t_0)-f(x,y,t_{-1}))^2$$

$$MAD(d) = \frac{1}{MN}\sum_{(x,y)\in R} f(x+d_x,y+d_y,t_0)-f(x,y,t_{-1})$$

<식2> MSD, MAD 최소비용 함수

4.4 매크로 블록과 서브 매크로 블록



(a)매크로 블록 (b)서브 블록

[그림 4] 매크로 블록 / 서브 블록

그림 4.(a)는 16*16의 기본 매크로 블록으로 기본 탐색 크기로써 우선 MAD의 결과 값을 통해 서브 매크로 블록 사용 여부를 결정한다. 블록 내부의 최적의 움직임 벡터가 임계치를 벗어난 움직임일 경우 서브 매크로 블록탐색을 통해 그림 4.(b)와 같이 0, 1, 2, 3의 4개의 블록에 벡터 값이 저장 되어 영상 보상 시 해당 블록에서 4가지 값을 동시에 보상 한다.

4.5 움직임 보상

기본적 움직임 보상과정은 미래 영상을 현재 영상보다 먼저 인코딩하여 정확한 움직임 정보를 토대로 움직임 보상 방법을 결정한다.

미래 영상의 매크로블록과 현재 영상탐색영역의 가장 일치하는 영역을 빼면 오차 매크로블록이 생성되며 가장 일치하는 영역의 위치를 벡터 값과 함께 전송한다. 인코더에서 오차매크로 블록은 코딩된 후 다시 복원 되어 가장 일치하는 영역과 더해져 매크로 블록을 재구성 하게 되는데, 이렇게 재구성된 매크로블록은 이후의 움직임 보상을 위한 참조 매크로블록으로 사용되기 위해 저장된다. 인코더와 디코더가 움직임 보상을 수행할 때 동일한 참조 영역을 사용하도록 하기 위해서 복원된 오차 매크로 블록을 사용하여 매크로블록을 재구성 한다.

움직임 보상을 위한 블록 사이즈가 작을수록 보다 나은 움직임 보상 결과를 얻을 수 있다. 그러나 블록 사이즈가 작아질수록 더 많은 탐색을 수행해야 하므로 연산양이 증가하여 전송해야 할 움직임의 벡터의 개수도 많아진다. 또, 비디오 영상에서 움직이는 객체가 16*16블록 영역을 따라 움직이는 경우는 많지 않기 때문에 움직임 보상을 위해 다양한 블록 크기를 사용하는 것이 보다 효과적이다.[6][7]

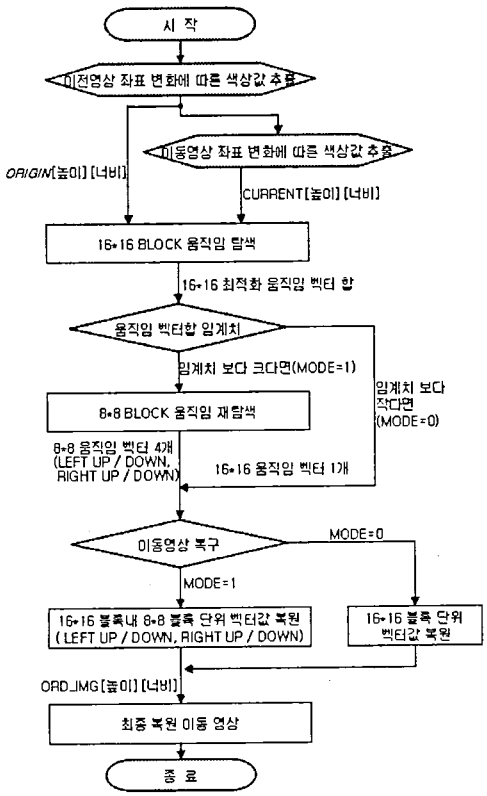
4.6 움직임벡터추출 프로그램 순서도

그림 5 순서도의 흐름은 두개의 row영상파일을 ASCII, BINARY 모드로 읽어 영상파일의 영상 값을 origin[i][j], current[i][j] 에 각각 저장한다. 최초 16*16의 블록 사이즈, -8~7탐색 범위로 전 탐색 블록매칭을 통해 최적의 움직임 벡터 값을 얻어낸다. 이때 해당 벡터 값이 임계치보다 작으면 현재의 움직임 벡터 값을 저장하며 해당 블록의 좌표(a, b)를 MODE[a][b]=0 으로 저장 하여 영상 보

상 시 해당 좌표에서 16*16블록의 벡터 값 1개로 영상을 복구할 수 있도록 한다.

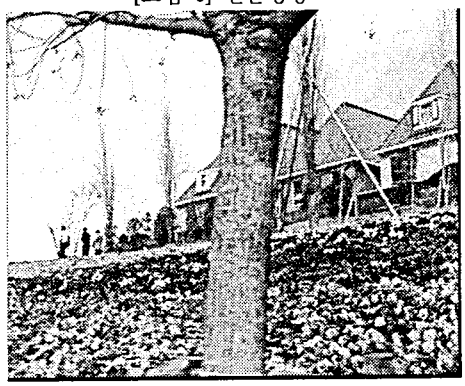
만약 16*16 전 탐색을 통한 최적의 움직임 벡터 값이 임계치보다 크면 해당 블록을 8*8블록, -4~3 탐색범위로 움직임 영상을 기존 16*16 블록의 1/4 크기로 재탐색 하여 더욱 정확한 움직임 벡터 값 4개를 얻는다. 이때 나오는 4개의 벡터 값은 16*16 블록을 기준으로 왼쪽 위, 왼쪽 아래, 오른쪽 위, 오른쪽 아래 영역으로 나누어진다. 16*16 블록의 해당 좌표(a, b)를 MODE[a][b]=1 으로 저장 하여 영상 보상 시 8*8블록의 벡터 값 4개로 영상을 복구할 수 있도록 한다.

영상 복구 방법은 16*16블록 단위로 좌표(a, b)를 증가시키며 MODE[a][b]의 값과 비교하여 복구 방법을 선택 한다. 선택된 MODE에 따라 순차적으로 이전 영상에 벡터 값을 대입함으로써 벡터 값만으로 최종 이동영상 영상 값을 ord_img[ ][ ]에 저장하여 row파일을 생성한다[8].

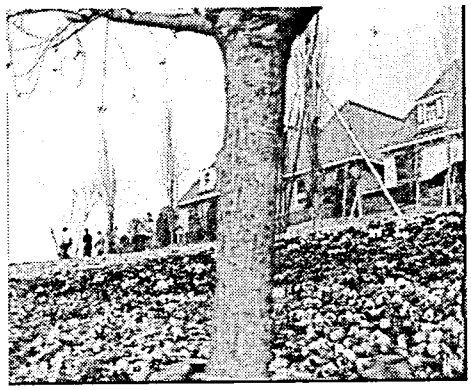


[그림 5] 프로그램 순서도

[그림 6] 원본영상



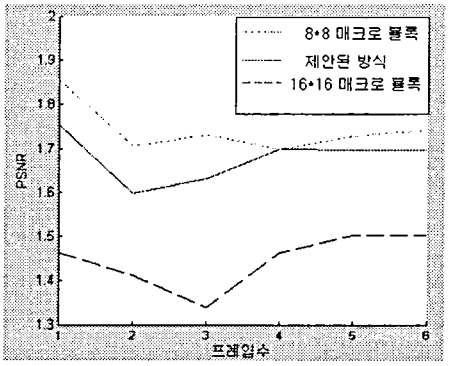
[그림 7] 참조 영상



[그림 8] 복원 영상

그림 6, 7은 이전 영상과 이동 영상이다. 이 두 영상을 이용하여 움직임 벡터를 얻어냈으며 얻어낸 움직임 벡터를 이용하여 그림 8을 복원하였다.

5. 결과



[그림 9] PSNR

그림 9는 8*8 매크로 블록만을 이용하여 움직임 벡터를 얻은 후 복원한 영상과 본 논문에서 제안하여 프로그램 한 방법으로 움직임 벡터를 얻은 후 복원한 영상 그리고 16*16 매크로 블록만을 이용하여 움직임벡터를 얻은 후 복원한 영상을 각 6프레임에 적용하여 얻은 PSNR 값이다. 그림에서 보면 8*8 매크로 블록을 사용하였을 경우 가장 좋은 결과가 나오나 압축해야 할 정보량이 많아 비효율 적이다.

6. 결론

영상 압축에 관한 연구는 계속적으로 이루어지고 있다. 그중에서도 최근 발표한 **H.264** 관한 연구들은 더욱더 활발히 연구되고 있다.

본 논문은 DMB용 움직임 벡터 추출 프로그램을 구현하여 16*16 기본 탐색 블록에서 움직임에 따라 가변적으로 8*8블록 탐색을 실행함으로써 단순히 16*16 블록 탐색만 하였을 경우 보다 영상 복구 능력을 향상 시켰다. 또한, 8*8 블록 탐색만 하였을 경우 불필요한 정지 영상에 대한 연산을 줄여 속도를 증가 시켰다. 이후 이를 바탕으로 움직임 벡터 추출 모듈을 VHDL로 구현한 후 성능 평가의 기준으로 이용이 가능하다.

7. 참고문헌

[1] 후지와라 히로시, 정제창 역, "최신 MPEG", 교보문고 1995.

[2] 이호석 · 김준기, "알기 쉬운 MPEG-2: 소스코드 해설", 홍릉과학 출판사, 2002.

[3] Iain E.G. Richardson, "H.264 and MPEG-4", WILEY, 2003.

[4] John Watkinson, "The MPEG Handbook", Focal Press, 2001.

[5] Fernando Pereira · Touradj Ebrahimi, "The MPEG-4 Book, 2002.

[6] Joint Video Team of ISO/IEC MPEG & ITU-T VCEG, "JVT-G050r1", 2003.

[7] http://iphome.hhi.de/index.php

[8] http://www.tnt.uni-hannover.de/js/project/jvt/